# CSC 372, Spring 2016
# Assignment 2
# Due: Friday, January 29 at 23:59:59

## Introduction

There's no programming at all on this assignment. It's a combination of web research (for two problems), creative thought, pondering, and a little writing.

For each problem you are to answer via a plain ASCII text file. Use an editor like Sublime, Vim, Emacs, Notepad++, etc. DO NOT submit Word documents, PDFs, Rich Text files, HTML documents, etc. As a double-check, your `.txt` files should look perfectly fine when displayed with `cat` on lectura, and the `file` command should show them as "`ASCII text`" or maybe "`ASCII English text`", or something similar.

Important: You'll be using a bash script on lectura to turn in your work. I recommend you give the submission process a try well before the deadline, in case you have trouble with it. See page 4.

## Problem 1. (6 points) `morefacts.txt`

When covering slide 24 in the intro set (http://www.cs.arizona.edu/classes/cs372/spring16/intro.pdf) I said a sentence or two about various languages of interest. For this problem I'd like you to find three languages that are not mentioned on that slide and tell me a sentence or two about each.

**I'll compile all the answers and post them on Piazza.** Follow this format for your answers:

(1) The full response for each language should be a single line of text.
(2) Begin the line with the language's name followed by the year it appeared and then a colon, followed by one or more sentences with whatever you want to say.
(3) End each line with an attribution that is either your name or "anonymous".

As examples of both the format and the sort of thing I'm looking for, here are three of mine, one with anonymous attribution. I'll use `cat` to display my `morefacts.txt` and then `wc -l` to demonstrate it's only three lines:

```
% cat morefacts.txt
Ada 1980: The DoD's attempt to have one language for military embedded systems,
instead of 450. -- William Mitchell
Java 1995: The most rapidly adopted language of all time.   In Spring 1997 I gave
one lecture in 372 about Java as a rising language; by Fall 1998 it was being taught
in 127A. -- William Mitchell
Scala 2003: Proof that Germans should stick to beer and BMWs. -- anonymous
% wc -l morefacts.txt
3 morefacts.txt
```

**Feel free to use Google, Wikipedia, etc., for research on this question** but needless to say, no posts anywhere soliciting ideas.

Above I say, "the year it appeared" but that's often subject to debate. Feel free to believe Wikipedia or go with other sources.

Just to be clear, you may use anonymous attribution to keep your classmates from knowing you wrote a particular entry but what you submit must be original, not something you found on the net.

**Problem 2. (6 points) `jp.txt`**

Slide 35 in the intro set raises the idea of the philosophy of a language. In a nutshell, I think of the philosophy of a language as what it treats as important, or not. For this problem I'd like you to identify three elements of the philosophy of Java.

**For this problem it's fine to brainstorm with CLASSMATES, Google for "what is the philosophy of java", etc., but what you submit must be stated in your own words.**

A piece of "low-hanging fruit" in Java's philosophy that I'm hereby prohibiting you to use is support for object-oriented programming. If it were not prohibited, here's what you might have said about that element:

> *Java supports the object-oriented paradigm by providing classes and inheritance. The "abstract" keyword allows classes and methods to be marked as abstract. The "static" keyword, although poorly named, supports the concept of class variables and methods.*

**Problem 3. (3 points) `measure.txt`**

Slide 27 in the intro set cites some attempts to measure language popularity. Some use simple measures, such as new GitHub repositories and job postings. The TIOBE index is more complicated. http://www.code2015.com was a simple tweet-based survey.

For this problem I'd like you to invent another simple way to measure language popularity. For example you might say, "Stand at the intersection of Speedway and Campbell and count programming language-specific bumper stickers." That's not a bad first thought, but I'd be worried about a dearth of data points and not be inclined to award that idea full credit.

Along with describing your idea, mention any weaknesses you see with it. For example, a weakness with code2015.com was that it wasn't widely known. Another is that such polls are subject to inflation by promotion of it within user communities. (Just for fun: see if you see any code2015.com results that seem way out of line. There was a code 2014.com, too, with more participation.)

**No web research or discussion with anybody else for this problem, please. It's just you and your brain on this one.**

Any ideas that make me say "Wow!" will earn a point of extra credit.

If you should find yourself completely blank for an idea 48 hours prior to the deadline, ask me for a hint.

**Problem 4. (2 points) `negative.txt`**

Java's `String.charAt()` method allows strings to be indexed from the left end of the string but not the right end of the string—`s.charAt(0)` is the first character, but we need to write something like `s.charAt(s.length()-1)` to get the last character. In contrast, many languages interpret negative string indices as being right-relative: `-1` is the last character, `-2` is next to last, etc.

For this problem you are to imagine you're designing a new language and you are currently considering whether to support negative indexing for strings. Present an argument that's either in favor of negative indexing, or against it.

**Problem 5. (1 point) `javarepl.txt`**

http://www.javarepl.com is a REPL for Java. Spend a few minutes experimenting with it and write a few sentences about it. You might talk about what you liked, what you expected to work but didn't, what you'd like to change, etc.

**Problem 6. (ZERO points) `quickrepl.txt`**

After working with languages that have a REPL available, you may find yourself really wanting a REPL when you go to learn a language that doesn't have one. One approach to a REPL is to integrate it with a language's implementation, but that typically requires a good understanding of that implementation. Another approach is a completely standalone REPL, such as provided by javarepl.com, but that can require a lot of code—`wc` shows about 8,000 lines in Java source files for that system.

For this problem your challenge is to come up with a quick approach to write a simple REPL for a language. Your quick REPL needn't be nearly as good as `ghci` but it should be able do the sorts of things you do when learning a language, such as evaluating expressions and showing types. Think about an approach that provides a lot of functionality for relatively little effort—something you could implement in a day, and not be late for dinner.

I'm not asking you to write any code for this problem; just write a description of how you might quickly produce a simple REPL for a language of your choice.

**Note:** This is a hard problem that requires some creativity and cleverness, so I'm making it worth zero points, but I'm curious to see who'll do it anyway! For this problem you are free to work in any groups of any size, but for me to possibly be impressed with your answer you'll have to specifically state that you came up with whatever you did all by yourself/yourselves—no Googling.

If you work in a group, all members may submit identical copies of `quickrepl.txt`, but it should include list of group members.

**Problem 7. Extra Credit `observations.txt`**

Submit a plain text file named `observations.txt` with...

(a) (1 point extra credit) An estimate of how long it took you to complete this assignment. To facilitate programmatic extraction of the hours from all submissions have an estimate of hours on a line by itself, more or less like one of the following three examples:

```
Hours: 6
Hours: 3-4.5
Hours: ~8
```

If you want the one-point bonus, be sure to report your total (estimated) hours on a line that starts with `"Hours:"`. There must be only one `"Hours:"` line in `observations.txt`. It's fine if you care to provide per-problem times, and that data is useful to us, but report it in some form of your own invention, not with multiple lines that contain "Hours:", in either upper- or lower-case.

Other comments about the assignment are welcome, too. Was it too long, too hard, too detailed? Speak up! I appreciate all feedback, favorable or not.

(b) (1-3 points extra credit) Cite an interesting course-related observation (or observations) that you made while working on the assignment. The observation should have at least a little bit of depth. Think of me

saying "Good!" as one point, "Excellent!" as two points, and "Wow!" as three points. I'm looking for quality, not quantity.

**Turning in your work**

Here's the full list of *deliverables* for this assignment

- `morefacts.txt`
- `jp.txt`
- `measure.txt`
- `negative.txt`
- `javarepl.txt`
- `quickrepl.txt` (zero points—it's optional!)
- `observations.txt` (for extra credit)

Note that all characters in the file names are lowercase.

Do not include your name, NetID, etc. in your `.txt` files—I like reading answers without knowing who wrote them.

The bash script `/cs/www/classes/cs372/spring16/a2/turnin` (on lectura) should be used to turn in your work. It creates a time-stamped tar file that contains the expected files and then uses the system-wide `turnin` program to actually turn in that tar file.

I recommend that you create a symbolic link named `a2` that references the `/cs/www/classes/cs372/spring16/a2` directory. Here's a command that creates such a link:

```
ln -s /cs/www/classes/cs372/spring16/a2 .
```

(Yes, that final argument, a dot to specify that the symbolic link be made in the current directory, can be omitted.) If you haven't worked with symbolic links, slides 134+ in my 352 UNIX slides, on the Piazza resources page, talk about them.

Assuming you've made that symbolic link, you should see something like the following when you run `a2/turnin`:

```
% a2/turnin
Warning: quickrepl.txt not found
======== contents of a2.20160118.143419.tz ========
-rw-r--r-- whm/whm         521 2016-01-18 14:00 morefacts.txt
-rw-r--r-- whm/whm         692 2016-01-18 14:00 jp.txt
-rw-r--r-- whm/whm         535 2016-01-18 14:00 measure.txt
-rw-r--r-- whm/whm         880 2016-01-18 14:00 negative.txt
-rw-rw-r-- whm/whm         188 2016-01-18 14:00 javarepl.txt
-rw-r--r-- whm/whm         353 2016-01-18 14:00 observations.txt
======== running turnin ========
Turning in:
 a2.20160118.143419.tz -- ok
All done.
```

Note that the first line of output, "`Warning: quickrepl.txt not found`", indicates that a deliverable was not found. We'll imagine that since that `quickrepl.txt` is a zero-point problem, the student elected not to do it.

After that warning, the contents of the tar file, named `a2.20160118.143419.tz`, are shown. Finally, the system-wide `turnin` program is run and its output is shown.

You can run `a2/turnin` as often as you want. We'll grade your final non-late submission.

We can add a `-ls` option to `a2/turnin` to show what's been turned in:

```
% a2/turnin -ls
.:
total 4
-rw-rw---- 1 whm whm 1878 Jan 18 14:34 a2.20160118.143419.tz
```

**If a submission is late, i.e., it shows a date of January 30 or later, it will be ignored unless you mail to `372s16` with some reason as to why it was late.**

**Miscellaneous**

Point values of problems correspond directly to assignment points in the syllabus. For example, a 10-point problem would correspond to 1% of your final grade in the course.

Remember that late assignments are not accepted, and that there are no late days; but if circumstances beyond your control interfere with your work on this assignment, there may be grounds for an extension. See the syllabus for details.